M: *packed array* [1..7] *of char*, ou
M: *array* [1..7] *of char*, ou ainda
M: *string* 7

dependendo do compilador usado.

Com caracteres (*strings*) poderemos utilizar as funções embutidas:

ord (c): devolve o número ordinal do caracter C no conjunto de caracteres utilizado.
chr (I): devolve o caracter com número ordinal *i*.

Assim

chr (ord (c)) = *c* e
ord (chr (i)) = *i*

além disso

$C_1 < C_2$ se e só se ord (c_1) < ord (c_2)

As funções pred e succ também podem ser usadas para caracter:

pred (c) = chr (ord (c) − 1)
succ (c) = chr (ord (c) + 1), se *c* é um caracter.

TABELA DOS CARACTERES ASCII

y \ x	0	1	2	3	4	5	6	7
0	nul	dle		0	a	P	`	p
1	soh	dc1	!	1	A	Q	a	q
2	stx	dc2	"	2	B	R	b	r
3	etx	dc3	≠	3	C	S	c	s
4	eot	dc4	$	4	D	T	d	t
5	enq	nak	%	5	E	U	e	u
6	ack	syn	&	6	F	V	f	v
7	bel	etb		7	G	W	g	w
8	bs	can	(	8	H	X	h	x
9	ht	em	)	9	I	Y	i	y
10	lf	sub	*	:	J	Z	j	z
11	vt	esc	+	;	K	[	k	[
12	ff	fs	,	<	L		l	|
13	cr	qs	−	=	M	]	m	]
14	so	rs	.	>	N	↑	n	¬
15	si	us	/	?	O	_	o	del

ord (c) é o número de ordem do caracter *c* no conjunto dos caracteres ordenados (usando a tabela ASCII, ord (c) = 16 **x* + *y*, se *x* e *y* denotam as coordenadas do caracter (*c*).

l) LÓGICO

PORTUGOL
lógico: P;
P ← *verdadeiro*;
P ← *falso*;

PASCAL
var P: *boolean*;
P: = *true*;

No PASCAL, considera-se

false < *true*; e é possível utilizar os operadores:

and	e
or	ou
not	não

m) VARIÁVEIS E CONSTANTES

PORTUGOL
real: RAIZ1, RAIZ2, PI;
RAIZ1 ← raiz (B **2 − 4 *A *C) − B;
PI ← 3.1416 {não varia em todo o programa};
RAIZ2 ← 0;

PASCAL
const PI = 3.1416;
var: RAIZ1, RAIZ2: *real*;

No PASCAL, é conveniente distinguir entre constantes e variáveis, por questões de otimização (economia) de memória. As constantes são armazenadas em binário com apenas o número de bytes necessários.

A sintaxe para constantes e variáveis é:

